- -> initialising different arrays
- ***-> zeros <- np.zeros(shape)***
  - -> for example, the shapes can be 5 <- an entire number
  - -> (rows, columns)
  - -> you can do nxmxd dimensions
  - -> another example is a matrix of all 1's

- ***-> ones***
  - -> you can also specify the datatype in the argument

```
In [42]: # All 1s matrix
         np.ones((4,2,2), dtype='int32')
```

  - -> a matrix with these dimensions, which is just made out of 1's

- ***-> you can also initialise matrices which aren't 1's of zeros***

```
In [44]: # Any other number
         np.full((2,2), 99, dtype='float32')

Out[44]: array([[99., 99.],
                [99., 99.]], dtype=float32)
```

  - -> create an array of values, with this shape
  - -> make the array just full of 99's and use this datatype
  - -> this specifies the amount of memory we want to give the element
  - -> array creation routines

- ***-> full_like***
  - -> np.full(a, 4)
  - -> return an array which is the same shape as the a array, and is only full of 4's

- ***-> to initialise an array / matrix of random numbers***
  - -> random decimal numbers
  - -> np.random.rand(4,2)
  - -> you pass in the integers for the shape
  - -> this returns an entire array of random numbers, which has a shape of 4x2
  - -> a.shape <- to pass in a shape

- ***-> if you wanted random integer values***
  - -> np.random.randint(7)
  - -> to return one random number up to 4
  - -> you can np.random.randint(7, size=(3x3))
    - -> for example, this is returning a 3x3 matrix full of random integers up to 7
    - -> using the documentation if stuck
    - -> every time you run the cell, it returns a new set of random integers

- ***-> if you want to do the identity matrix***
  - -> np.identity(3)
  - -> this returns a 3x3 identity matrix
  - -> you can also repeat an array

    - -> np.array([1,2,3])
    - -> np.repeat(arr,3,axis=0)
    - -> you can also make this a two dimensional array

- ***-> question***

What will the following code print?

a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(np.full_like(a, 100))

[[100 100 100 100 100]]

[[100 100 100 100 100] <- This one
[100 100 100 100 100]]

[[1 2 3 4 5]
[6 7 20 9 10]]